DOCUMENT RESUME

ED 264 249 TM 850 483

AUTHOR Shale, Douglas G.; Milinusic, Tomislav O.
TITLE A Computer Support System for the Entry and Analysis of Questionnaire Data.
PUB DATE Mar 85
NOTE 24p.; Paper presented at the Annual Meeting of the American Educational Research Association (69th, Chicago, IL, March 31-April 4, 1985).
PUB TYPE Speeches/Conference Papers (150) -- Reports - Research/Technical (143)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Computer Oriented Programs; *Course Evaluation; Data Analysis; *Databases; *Data Processing; Distance Education; Foreign Countries; Higher Education; Information Retrieval; Information Storage; *Questionnaires
IDENTIFIERS Athabasca University AB

ABSTRACT

This paper describes a computer support system that eliminated many of the problems associated with the usual methods of transcribing and analyzing questionnaire data. The system was created to support the course evaluation system at Athabasca University, a distance education university in Canada. The courses evaluated were all home study courses, and they varied significantly in design and with respect to the kinds of instructional materials they incorporated. Consequently, the evaluation questionnaires were custom tailored to each course, and little standardization was possible among courses. The sheer volume and variety of questionnaires required that some form of computer support be used. Within two years, the system supported the evaluation of 42 different courses requiring 125 distinct questionnaires. Software was written in the C programming language to run on a VAX 11/780 under the UNIX operating system. This configuration allowed for text editing features, specific and global scanning, and global alteration of sets of codes. Technical aspects of the system, as well as its limitations, were discussed in detail. (LMO)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED264249

A COMPUTER SUPPORT SYSTEM FOR THE ENTRY AND ANALYSIS OF QUESTIONNAIRE DATA

DOUGLAS G. SHALE
THE UNIVERSITY OF CALGARY

TOMISLAV O. MILINUSIC
ATHABASCA UNIVERSITY

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☑ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

D. G. Shale

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

A paper presented at the American Educational Research Association Annual Meeting, March 31 - April 4, 1985; Chicago, Illinois.

TM 850 483

ABSTRACT

A COMPUTER SUPPORT SYSTEM FOR THE
ENTRY AND ANALYSIS OF QUESTIONNAIRE DATA

Douglas G. Shale
The University of Calgary

Tomislav O. Milinusic
Athabasca University

The conversion of information gathered through questionnaires to machine readable forms is typically a time consuming, costly task fraught with opportunity for introducing inaccuracies as information is transcribed from one medium to another. A variety of procedures and constraints are often invoked to simplify the transcription task and consequently minimize these problems. Often such standardization places additional burdens on questionnaire designers, respondents, and data entry personnel. Such burdens are more appropriately and efficiently accommodated through computerization. This paper describes a computer supported system that eliminates many of the problems arising from the usual methods of entering and analyzing questionnaire data. The paper also describes an application of this system.

A Computer Support System for the Entry and Analysis of Questionnaire Data

The conversion of questionnaire information to machine readable form is typically a time consuming, costly task which needs to be monitored carefully because of errors that may be introduced as information is transcribed from one medium to another. A variety of procedures and constraints are often invoked to simplify the transcription task and consequently minimize the potential for introducing error and the complications arising when the data are to be processed. These procedures usually require additional manual effort in formatting the questionnaire and in transcribing and verifying data -- effort which can be substantially redundant if one subsequently finds that the level of reliability in the transcription process is reasonably high. The redundancy, however, is a necessary evil associated with manually based procedures inasmuch as the reliability of the procedures can only be ascertained by duplicating them.

Much of the standardization of format forced on questionnaires and on the transcription of questionnaire information to machine readable data exists to render data entry and data analysis tasks more manageable and to check the accuracy and validity of the data. However, it is often unnecessary and may be damaging to a study's objectives to force questionnaire respondents or data entry personnel to bear the burdens accruing to this standardization. The computer is much better at sorting out such matters as pre-columning, data validation and error checking. Furthermore, using a computer support system can confer additional advantages that may not be realized through manual procedures -- for example, compiling and coding responses to open-ended questions, and comprehensive editing through text processing. Perhaps more importantly, using computer support allows for more customized design of questionnaires. It would no longer be necessary to administer the same general questionnaire over a variety of situations simply to render data entry and analysis more manageable.

The objective of this paper is to describe a computer support system that eliminates many of the problems associated with the usual methods of transcribing and analyzing questionnaire data. The paper also describes an application of this system.

A Review of Manually Based Procedures

In the earlier days of data processing, questionnaire data were often transposed by hand to coding sheets which then served as an instruction set to key punch operators regarding what alphanumeric characters were to be entered in what card column fields. The person doing the transcribing would be guided by a set of coding roles which specified: (i) what alphanumeric equivalents were to be assigned to what questionnaire responses; and (ii) how to assign unique card column fields to accommodate the alphanumeric data. Many of the coding rules resulted from constraints inherent in using a Hollerith card. For example, the 80 column capacity of the card often lead to considerable effort to conserve columns (perhaps through more rigid question design or the use of hard-to-manipulate multiple punches) so that only one card would be required. When multiple cards were required, extra (and often redundant) effort was required to duplicate identification information so the card set corresponding to a questionnaire could be identified. Continuation information would also have had to be added to specify the number of cards in a set and the order in which they should appear. In addition, extra error checking would then be required to ensure that card sets were complete and cards were in the proper order.

While mark sense cards and optically scanned forms are appropriate and efficient for some types of surveys, they seem not to have been widely used because they require specialized production methods and specialized, expensive equipment to process. Because of their limited applicability, these methods of data entry are not specifically considered in this discussion.

A major advance in the transcription process was the introduction of the concept of "pre-columning" questionnaires. (Some authors, for example Erdos, 1983, consider pre-columning to be part of the pre-coding process). In essence, pre-columning questionnaires simply incorporates into the questionnaire the data field information that would otherwise have been specified through use of a coding sheet. Consequently, data entry personnel can key punch directly from a questionnaire thus eliminating the manual operation of transcribing questionnaire responses to coding sheets. This eliminates considerable error associated with this process. Although there are a variety of formats that may be used to pre-column a questionnaire

(see, for example, Boser and Heathington, 1984), all require column specification information to be printed on the questionnaire. Judiciously used, pre-columning is not considered to have an effect on people's predisposition to respond or on the manner in which they respond. Sudman and Bradburn (1983; pp. 258) state, "There is no evidence, however, that pre-coding and pre-columning bother respondents on self-administered questionnaires." However, Boser and Heathington (1984) do cite some evidence of respondent preference for a particular type of pre-columning format.

Although pre-columning questionnaires saves the time and expense of transcribing responses to coding sheets (as well as eliminating an additional source of error), the constraints enumerated earlier of Hollerith cards and their limited fixed-field format remain. Backstrom and Hursh (1963; pp. 162-164) among others, present a checklist of coding hints that characterizes the considerations that typically arise in this regard. In summary, these limitations are:

- cards, physically, are an awkward medium to handle and use. The more the card set must be handled the more potential there is for introducing error through inadvertent shuffling of card order or through data input due to card damage or card reader malfunction.
- cards need to be counted to make sure every questionnaire has been entered and that none has been entered twice. Cards must be checked for off-setting errors by sorting them in serial order (especially if there are multiple cards) and checking for duplication of serials and for missing numbers.
- because using multiple cards makes data processing and analysis more complicated, data are often compacted to fit within the 80 columns on a card. Occasionally, this will require that multiple punches be used, possibly complicating the data analysis in other ways. Most certainly it renders visual scanning of the data on the cards (either for verification or alteration) more difficult.
- retrieving particular cards to be reviewed or amended can be time consuming and difficult.
- amending errors requires redundant effort because entire cards have to be re-punched. If an error is general, then much work has to be re-done and errors fixed individually.

- the use of multiple cards per case requires that redundant information be carried on each card (minimally an identification number). In addition, card continuation information must also be introduced. In both cases, the additional information added requires that additional error checking occur and adds to the volume of data to be handled.
- the matching of data to specified fields on a card is managed through the human interface (the data entry personnel). Although good pre-columning design and training can minimize the error rate associated with this process, additional effort and care must nonetheless be expended.
- leaving too few columns in pre-columning a data field results in time consuming manual fix-ups. The problem can be exacerbated when open-ended questions are to be classified (as would be the case, for example, if one expected that fewer than 10 categories would be required and found that more were necessary. Questions requiring multiple responses and questions requiring respondents to specify a number can also lead to this kind of dilemma). The standard fix-ups usually are introduced through an editing process and result in a more complicated set of key punching directions.
- the form in which data are collected and converted to machine readable form may not be the optimal form in which to store and analyze the data. For example, many questions may be asked of a respondent, but few responses may result. This usually means that many null fields must be entered, increasing the number of cards to be used and the amount of data processing and storage required. One approach to dealing with the data entry problem is to instruct the data entry personnel on procedures for skipping fields (which automatically defines what blanks will mean). However, this is still awkward and doesn't resolve the problems of multiple cards and unnecessary storage and processing requirements.
- digits with fewer place values than columns reserved for their entry must be right-justified. Otherwise, zeroes may be added to

the right of the number during data input to fill out the field, resulting in values that are incorrect by powers of ten.

- indicating blanks, zeroes and values for missing data can be problematic, depending to a large degree on the software to be used for data analysis. Once a choice is made on the coding rules for blanks, zeroes and values, changing the rules (perhaps to accommodate the need to use a variety of software packages) requires a change to the cards and ensuing considerable inconvenience and extra labour.

Verification

Errors resulting from keypunching cards are usually checked through a process called "verification." In this process, a verifier operator repeats a keypunch operator's work on a machine called a "verifier." The verifier indicates any discrepancy between what the two data entry personnel have keypunched. As Erdos (1983, pp 191) points out, "For all practical purposes verification means doubling the time and cost of the keypunching operation."

While it is important to eliminate errors that may be introduced through keypunching, evidence cited by Erdos (1983) indicates that the amount of error introduced by qualified, experienced operators may be minor. One hundred per cent verification will always require the independent double entry of data. However, given that the rate of keypunching error seems typically to be small, instituting completeness and consistency checks may constitute "sufficient" verification.

Completeness and Consistency Checks

Good study design, a conscientious awareness of the pitfalls to be found in conducting the study, and a rigorous editing process will alleviate many of the "fatal" errors that can befall a survey study. However, data verification is still essential for ensuring that data are accurately transcribed. In particular, editing and verification are essential for detecting and correcting invalid code values (codes which are not members of the computer's character set) and illegal codes (codes not acceptable to

analysis programs to be used). However, there remain other sources of error that can yield less than obvious incorrect results.

Sonquist and Dunkelberg (1977; pp 197) have provided a list of some of these:

> "The existence of code values which are valid and legal, but which are not in the code for the variable in question. For example, a 7 recorded for a variable containing data that should be coded 1 (yes) or 2 (no) would be termed wild."
>
> "Inconsistencies in the data. For example, the response to a yes/no question about owing debt is coded "no," yet a non-zero value is recorded in the field containing the amount of debt owed."

Detecting wild values and inconsistent data by inspection can be very difficult, and often impossible. Sonquist and Dunkelberg (1977) and Erdos (1984) both advocate a cleaning process in which consistency checking is a major element. If the checking is to be done manually, then as Sonquist and Dunkelberg (1977; pp 198) point out: "If a study is very complex, it may be impractical to check for all possible inconsistencies." They further observe; "When checking is done on the computer, it is usually accomplished quickly; but, even here, large amounts of time can be consumed in the process of looking up and correcting inconsistencies when possible." Some portion of this time must unavoidably be invested in thinking about the kinds of inconsistencies that might occur and in devising appropriate checking procedures. If the checking is to be done by computer, then some time and effort will go into the programming. If it is not possible to use the same software to do the consistency checking as is used for the compilation and analysis (for example SPSS, PSTAT), then custom programming will have to be done and additional effort will have to be expended simply to accommodate the disparate software being used. Moreover, such checking software is run as an "add-on" activity after data entry has been completed. This means that an additional cycle is required to find and fix wild values and inconsistent data. Ultimately, reconciliation of errors

can only be done through the editing process. However, considerable efficiency could be achieved if the checking, flagging and fix-up could occur as data are being entered.

Handling Open-Ended Questions

Sonquist and Dunkelberg (1977; pp 86) have described the conditions under which the necessity for this type of question arises, and these conditions need not be repeated here. They point out that, in general, open-ended questions will be used when the researcher needs to ask a general question which by its nature will evoke unanticipated answers. In some cases, the answers cannot be anticipated to any reliable degree (despite pretesting). Occasionally, respondents will be asked to provide some specific numerical value, and in other cases only a partial categorization of responses may be possible. In this latter case, respondents are often given a category of response designated "other," and are requested to describe what their "other" response is.

In any event, such responses will be reviewed and classified as part of the editing process or ex post facto to data entry. When such questions are embedded in the body of a questionnaire, they are particularly problematic to accommodate in data processing. Insufficient space might have been reserved on the computer cards to receive the data once the open-ended questions have been coded. In addition, editing and coding require shuffling through the pages of each questionnaire and reviewing all the responses to a single question -- a procedure which can be very awkward and time consuming. For these reasons, designers of questionnaires will often place wholly open-ended questions at the very end. However, this would not be advisable if the result is to damage the logical flow of the questioning -- and usually it's not possible for those partially open-ended questions containing entries under an "other" category. Furthermore, it only partially ameliorates the problems of physically sorting through all the questionnaires and categorizing the open-ended responses using some manually based procedure.

Computer-Based Support Systems

For anyone with reasonably good programming skills, it is a relatively straightforward task to prepare an on-line, interactive computer program that will: (i) issue prompts to a data entry person indicating which response to which question should be entered; (ii) check on the validity of the data entered by ascertaining that it is the "correct" type of data (either alpha or numeric), and that the value is in the range of valid responses; and (iii) direct the data entered to an electronic file where it is stored in a specified location thereby creating in electronic form the conceptual equivalent of a deck of punched cards. It is also reasonably straightforward to design the system so that the text from open-ended questions can be entered at the same time that the closed-ended responses are (with the open-ended text occurring at any point in the questionnaire), and have this text routed to a separate file of text material that can be shuffled to whatever order is desired and viewed on the screen or printed out.

Systems already exist that essentially do this. Sudman and Bradburn (1983; pp 258) refer to a Computer Assisted Telephone Interviewing (CATI) system that seems, in principle, to be able to do tasks (i) and (iii). Reference will be made later in the paper to application generators.

Clearly, such a system would represent a major advance over the shortcomings and inconvenience associated with conventional, manually based data entry procedures. In particular, so many of the conditions and restrictions inherent in using punched cards would not exist or would be less onerous. Consistency checking and data cleaning could occur concurrently with data entry, greatly facilitating the editing tasks associated with finding and correcting errors.

The effort required to design this kind of support system is well worthwhile when one is doing a major survey study. However, this investment of labour will be prohibitive for small scale surveys. A solution to this problem is to build a computer-based system (a type of "application generator" if you like) to support the creation of such a prompting program. This is quite a sophisticated programming task that requires considerable effort to develop. However, once available, this general program

can be used repeatedly by someone with modest computer skills to quickly design a wide variety of customized prompting programs. Once the prompting program is available, data entry can be handled by anyone with some typing skill.

For convenience, the general " application generator" program will be referred to in this paper as the generalized prompt builder. The specific program built using the generalized prompt builder will be referred to simply as the prompt (or prompting) program.

Genesis of the System

The system was created to support the course evaluation system at Athabasca University (A.U.), a distance education university in Canada. The courses evaluated were all home study courses. In general, the courses vary significantly in their design and with respect to the kinds of instructional materials they incorporate. Consequently, the evaluation questionnaires were custom-tailored to each course, and little standardization was possible among courses. Each course typically required two or more questionnaires, with differences also occurring amongst these. With the number of courses involved and multiple questionnaires required for each course, there were a large number of different questionnaires that had to be handled. The sheer volume and variety of questionnaires required that some form of computer support be used. Within the first two years of its implementation (1979-1981), the system supported the evaluation of 42 different courses requiring roughly 125 distinct questionnaires. In that some period of time 1,065 questionnaires were processed.

Matters were further complicated because: (i) the number of completed questionnaires of one type was typically quite small (less than 50), and (ii) the questionnaires were returned in dribs and drabbles over extended periods of time (in some cases for as long as 3 or 4 years); however, evaluation reports had to be compiled on an irregular, occasional basis depending upon the particular requirements of a course and on the availability of data. In addition, there was only limited staffing available: about one-fifth of a professional staff member's time (to look after the conceptual design of the evaluation procedures and producing the

questionnaires), and about one-quarter of a clerk's time (to process, enter and compile questionnaire returns and to prepare summary reports). It was very apparent from the start that a computer support system would be absolutely necessary if there was to be a centralized course evaluation function at all.

When a new questionnaire form is received, the clerk invokes the generalized prompt building command to create a customized prompting program. The way in which this is done and the specifics for doing it are described in the next section. This prompt program is then used in an interactive fashion to support the direct entry of questionnaire data into computer files.

As the data are entered, validity and range checks are carried out as appropriate and as prescribed in building the prompt program. Errors are fixed as they are detected by the system, or their location is noted for later correction if required. The program directs the entered data to specific locations in electronic files for storage. Countable data are routed and stored in a data file. Text information which is entered as it is encountered in the data entry process, is routed to a text file. Each questionnaire is given an identification number when it is received. This ID number is the first data point entered for each questionnaire and serves to identify the lines of data stored in the data file. Moreover, when the response to an open-ended question is entered, the ID number is automatically appended to the text as it is stored in the text file. This ensures that there is a link preserved between an individual's responses to the closed and the open ended questions.

The software has been written in the C programming language to run on a VAX 11/780 under the UNIX operating system. Running under this configuration has bestowed additional benefits. In particular, we have been able to capitalize on the very powerful text editing features supported under UNIX to scan specifically and globally for data entries that need to be altered, and to make changes accordingly. In addition, if extra columns or fields need to be added to the data file structure after data entry has begun, the over-all structure of the data file can be managed through this facility. Editing can also be used to globally alter sets of codes -- as might be the case for blanks, missing values, and no responses.

As the system is currently constituted, compilation and analysis of the countable data are done by linking the data file to an appropriate SPSS program. In the case of the open-ended responses, the program groups together all responses to a particular question. This material can also be viewed, re-arranged, or amended interactively using text editing. Once this text is in the desired form, it can be printed out in whatever format is desired using text formatting commands.

Having all the responses to an open-ended question grouped together greatly facilitates categorizing and coding them. Once codes have been assigned, the data file can be opened using the text editor and the codes inserted. This is currently a manual task. Although this function could be automated, the amount of effort required to do so is not merited given that it can be quickly done manually.

Validity and range checks are specified through the generalized prompt program and become part of the prompting program. Automatic justification (either right or left) of all numbers can also be built in at this time, as can provisions for skipping fields depending on a response given previously. Although it is expected that the validity and range checking will remedy most of the "keypunching" errors that might occur, a verification mode can be specified through the generalized prompter. In verification mode, the data entry person is required to re-key the data from a questionnaire immediately after initial entry. Discrepancies are signalled by the terminal and the prompt program expects to be told what value ought to be entered. The verification mode routes the data entry person around open-ended questions so that text need not be re-entered.

As additional questionnaires are received, data derived from these can be easily added to the existing data file by opening the data file using the text editor. The prompt program will then direct newly entered data to the end of the appropriate data set and the data entry person simply proceeds as before. Data sets and text files for different questionnaires are kept as separate and distinct files. The prompt program, itself, can be stored indefinitely as a computer file in a directory with other prompt programs.

Technical Description of the System

The principal characteristics of the system are defined by the nature of the UNIX operating system, the generalized prompt-building program, the prompt file, the actual prompt program, and the text manager (Figure 1). Although the system could exist outside of UNIX in a different form and retain the same conceptual framework, it would not possess the flexibility and power that UNIX provides to it in several areas such as text-formatting, file-sharing and in other utilities native to UNIX.

UNIX is fast becoming a de facto standard operating system in the mini and supermini computer range. This powerful operating system has capabilities that are inherently suited to most of the computing needs of small to medium sized operations. Its chief attribute is the portability of the operating system across the whole range of computer sizes, from the 8 bit micros such as the IBM PC, to the 32 bit supermini, or to supercomputers such as the CRAY-2. The other two principal attributes of UNIX are its multi-tasking capabilities, where several processes or computer tasks with specific instruction can be executed simultaneously by any one user, and the multi-user aspect of its nature whereby many users can be signed on simultaneously and can access the same set of programs without conflict in resource or program allocations. A critical aspect of UNIX is the tree-like directory system that maintains all the files in the system. Any one user may access any other file in the system (provided permission is granted). This has direct relevance to the multi-tasking and multi-user nature of a shared resource such as the generalized prompter, the prompt builder and the text formatter.

The first program to be activated in a typical process of entry and analysis of questionnaire data is via the generalized prompt builder. This piece of software helps create a file that is used in subsequent questionnaire data entry operations. It creates the prompt menu for the prompt program. Its principal uses are two fold: (i) to provide appropriately placed prompts for the prompter; and (ii) to make sure that the data being entered meet certain defined criteria and to separate the data into two logical files according to their type. The first task essentially is an orchestrated set of questions that are input once, and displayed within the

prompter as many times as needed. The second task is at the heart of the system, and defines and verifies the nature of the input data.

The prompt file produced by the prompt builder specifies the conditions under which each element of data is collected. Both alphanumeric data and wholly numeric data are differentiated through it. The prompt builder program, when initiated, requests a file name where the prompt file is to be saved. If this already exists, it is appended to that prompt file. This is useful if the original prompt file's creation process had to be interrupted several times for some reason.

Having defined the name of a new prompt file, the prompt builder program then initiates a series of questions in a menu driven format. Thirteen fields define a prompt file procedure. However, only some of them are required depending on the type of data anticipated . The fields are as follows:

1. The text of the prompt is the line that will be displayed on the screen to solicit entry of a particular piece of questionnaire data be it numeric or textual. For example, the prompt may consist of a question number or it may be a mnemonic.
2. The mode field which contains a choice of four types describes the data's particular type, i.e. is it numeric data, textual or alphanumeric type of reply to a question, or a reply with more than one answer? A special "one time only" prompt is also elicited at the beginning of the program to get the questionnaire's ID from each questionnaire.
3. The type of field descriptor defines the Mode field in terms of whether it is of fixed length or variable length. This applies to both numeric and textual data.
4. If the type of field descriptor is of the fixed length variety, then its column width is requested at this stage.
5. If the data are fixed, then a further request as to whether it is to be left or right justified is prompted.
6. The minimum number for a numeric entry is requested.
7. The maximum number for a numeric field is requested.
8. A request is made for a decision as to which of the two files (i.e. the main or numeric file, or the textual file) the data entered should be saved in.

9. A separation character is requested if a variable length data is entered and its destination file is the main or numeric file.
10. A special string (any number of letters) is appended after the questionnaire I.D.
11. A termination string is requested at the end of each text record.
12. In textual data type entry, this field will convert the data entered to upper-case or lower-case as required.
13. If in numeric mode it is desirable to verify each entry, then an option may be implemented.

Figure 2 illustrates the flow of the generalized prompt-building program in some detail.

After a completed prompt file has been created, it is saved and can be accessed again through the second stage of the procedure involving the prompter program.

The prompt program repetitiously provides prompts for questionnaire data entry, and does logical and range verification after each datum entered, as outlined in the prompt file. It is menu driven, so that efficient data entry can be accomplished. Errors in data entry are flagged interactively. Text portions of the replies are automatically channeled to the text file that is created by the system. Numeric data are delimited and channeled to the appropriate numeric file for further processing by any of the statistical packages available on the system.

The two files mentioned above can be created and named individually or can be appended to previously created files for the same questionnaire. The text file is usually run through a UNIX spelling checker utility and all spelling mistakes are automatically and globally corrected. The textual file is an ASCII file and can be edited by any number of editors available on the system. The numeric files as well can be so edited.

After all questionnaire data have been saved into the two files, the numeric file is then processed conventionally via a "canned" package such as SAS, SPSS, S, BMDP etc. The text file is specially processed via the text manager, the third program in the suite.

The text manager is truly a product of the UNIX environment, in that it is trivial to implement and uses one of the most outstanding features of

UNIX - its text processing capabilities. The text manager sorts the text file according to one of two options; either by ID, or by question number. The sorted file is then converted to a special text-processing input format that uses a set of control macros called the NROFF macros. This new file is now ready to be sent to any type of output device, including the CRT screen, regular dot matrix printer, impact printers, laser printers and even to a phototypesetter without any other intervention. The parameters defining the output format can themselves easily be fine tuned should the need arise for changing the particular way the results are to be presented. Various additional textual material, such as the heading, footings, and other information are options requested by the program. It is also possible to integrate the original questions with the replies.

The flexibility of the text manager, like the other two pieces of software, owes much to the structure of the C language. C is a high level language which displays system and machine language level affinities. It is a structured language with precise syntax and powerful input/output control structures.

The text manager is essentially the integrator of textual data. The entire procedure of questionnaire analysis especially of textual material thus becomes much more amenable to computer processing. Artificial Intelligence (AI) type of programming may be applied to such textual data using a symbolic language such as LISP. The text manager is a very small step towards organizing data as a precursor to the AI process.

Discussion

In earlier sections of the paper, the limitations and shortcomings of the usual card-based, manually supported data entry process were described. It was also suggested that many of these limitations and shortcomings could be overcome by a computer supported system - and so they can. However, while the system described in this paper represents an advance in this direction, as it is currently constituted it does not do everything it might do, as well as it could. In particular, building in consistency

checks is problematic and currently would require direct programmer intervention to augment the prompt file. The generalized prompter does not currently facilitate this function.

In addition, although the system has the facility for converting the most conveniently entered alphanumeric character data to a form more amenable to computer analysis (for example, Y may be entered for a "yes" response, but "1" may be inserted in the data file), the system does not yet pack data as efficiently as possible. This means that while the system makes it very quick and easy to enter multiple null fields, it still stores substantial null fields in the data matrices thereby unnecessarily increasing storage requirements and costs. As the system has been applied in the course evaluation system it was designed to support, it is more economic to tolerate this waste than to undertake the additional developmental work to remedy the problem.

Furthermore, while the system greatly facilitates using open-ended questions and compiling and analyzing them, categorizing these responses and coding them still requires that someone work through the responses to convert them to a data form the computer can handle. However, as noted in the previous section of the paper, there is considerable potential for development in this area. Erdos (1983; pp. 186) has described an attempt to apply Artificial Intelligence to analyze textual responses through what he has called "concept coding" (see, also Frisbie and Sudman, 1968). Stated simply, in concept coding, the computer would do a context analysis of text according to the rules that would be invoked by an editor doing the classification and coding. A simple implementation of the concept could be achieved through key word specification and key word searching. The flexibility and power of the text processing facility under UNIX are such that developmental work in this direction could easily be undertaken.

Finally, the system <u>per se</u> does not really support data analysis. This has to be undertaken as a separate activity and, depending on the "canned" software package used, requires varying degrees of expertise and effort. Integration of these functions would be a great improvement.

As mentioned earlier, there are other kinds of computer support facilities available to do some of the same kinds of things as the system described here can do. One approach in particular should be mentioned - namely, using what Stevenson and Walleri (1982) have called "application

generators" and what are elsewhere often called "relational data bases." Stevenson and Walleri (1982) allude to their use of an application generator package called INFO (developed by HENCO, Inc. of Wellesley, MA.) to support direct data entry from questionnaires into specially designed screens. Kruglinski (1983) describes how to use a relational data base called CONDOR to do essentially the same thing.

It is considered beyond the scope of this paper to attempt a comparative analysis of the relative merits of using the system described in the paper or using application generators. However, there are obvious features to each general approach that would a priori predispose a researcher to choose one and not another. In particular, application generators are much simpler to use because they are packaged "turn-key" systems. In addition, application generators are available for micros thus conferring all the convenience and efficiency that this usually implies. On the other hand, the configuration of the AU system - while admittedly more complex - has considerably more "power" and more potential for subsequent development.

These considerations notwithstanding, experience with the AU system has conclusively demonstrated its effectiveness. Without such a system, the extensive course evaluation activity it supports simply could not continue to exist in its present form. Standardization of questionnaires and procedures would have had to be imposed to such a degree that the value of the information obtained would have been significantly diminished.

Acknowledgements

Developmental programming work on the system was done by Don Cowper who was then a member of Computing Services at Athabasca University. The authors also wish to acknowledge their use of documentation prepared by Don Cowper, Petra Hammer and Chris Swann.

References

Backstrom, Charles H. & Hursh, Gerald D.; Survey Research; Northwestern University Press, 1963.

Boser, Judith A. and Heathington, Betty S.; "Instrument Formatting with Computer Data Entry in Mind." Paper presented at the Annual Meeting of the American Educational Research Association, New Orleans, April, 1984.

Erdos, Paul L.; Professional Mail Surveys; Robert E. Krieger Publishing Company, Malabar, Florida; 1983.

Frisbie, Bruce & Sudman, Seymour; "The Use of Computers in Coding Free Responses," Public Opinion Quarterly, Vol. 32, No. 2, pp. 216-232, Summer, 1968.

Kruglinski, David; Data Base Management Systems: A Guide to Microcomputer Software; Osborne/McGraw-Hill, 1983.

Sonquist, John A. & Dunkelberg, William C.; Survey and Opinion Research: Prentice-Hall Inc.; Englewood Cliffs, New Jersey; 1977.

Stevenson, M. & Walleri, D.; Application generators: Their impact within Institutional Research and Beyond. Paper presented at the Twenty-Second Annual Forum of the Association for Institutional Research, Denver, Colorado, May, 1982.

Sudman, Seymour & Bradburn, Norman M.; Asking Questions; Jossey-Bass Publishers; San Francisco; 1983.

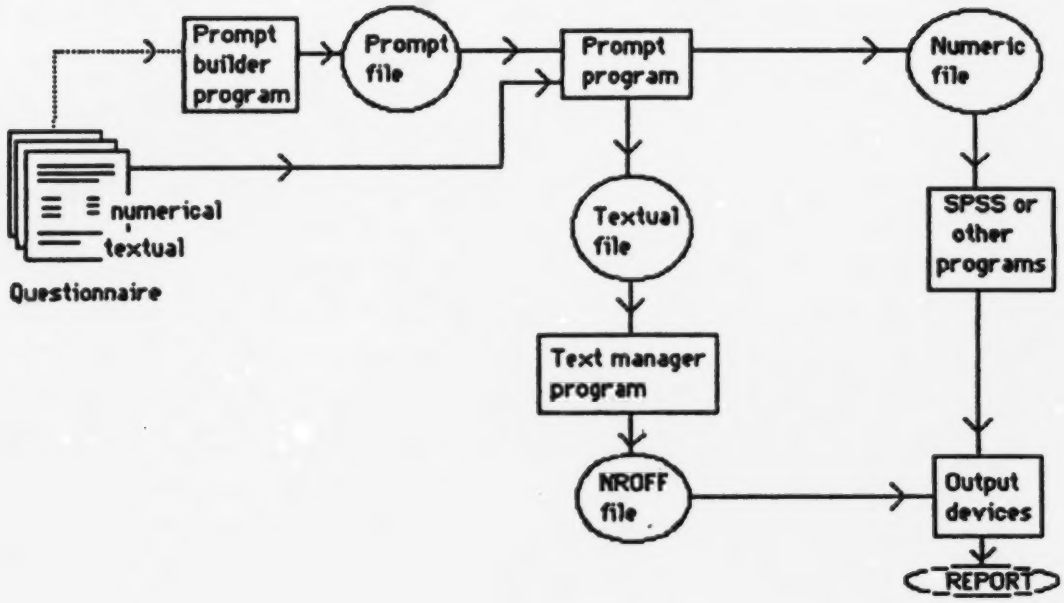

Figure 1: Schematic of the Computer Support System for Questionnaire Data Entry

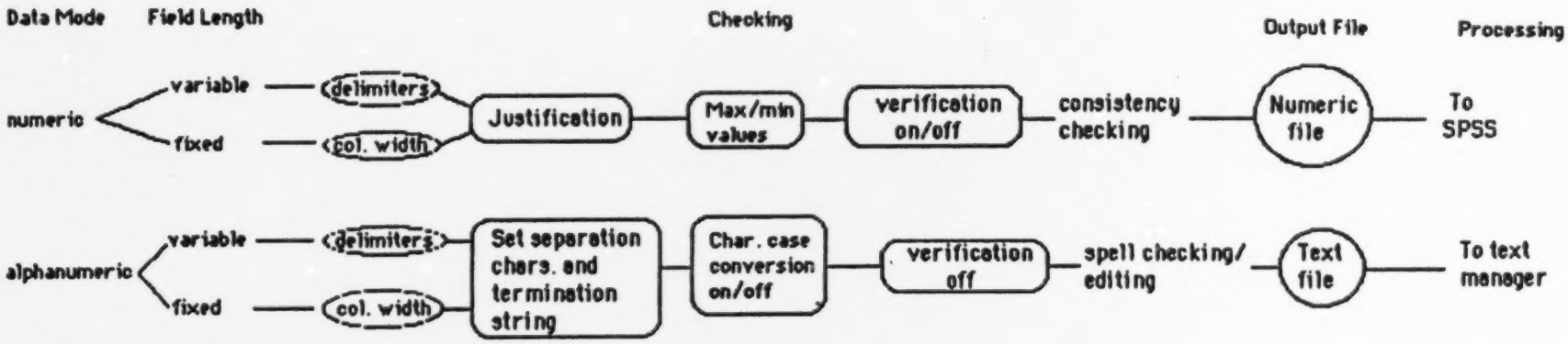

Figure 2: Flow Chart for Prompt Program

BEST COPY AVAILABLE